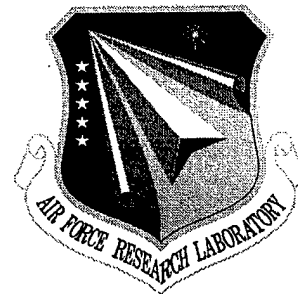


AFRL-IF-RS-TR-1999-251

In-House Report

December 1999



X WINDOWS-BASED INTERACTIVE TEST PATTERNS FOR OVERLAID STEREOSCOPIC AND TILED DISPLAYS

**Peter A. Jedrysik and Richard H. Sweed, AFRL
Robert VanPelt, Litton PRC**

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

**AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK**

DTIC QUALITY INSPECTED 2

19991227 033

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

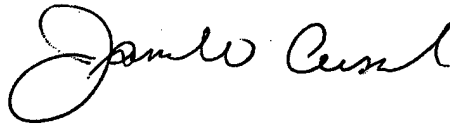
AFRL-IF-RS-TR-1999-251 has been reviewed and is approved for publication.

APPROVED:



STEVEN D. FARR
Chief, C4ISR Modeling & Simulation
Information Systems Division

FOR THE DIRECTOR:



JAMES W. CUSACK
Chief, Information Systems Division
Information Directorate

If your address has changed or if you wish to be removed from the Air Force Research Laboratory Rome Research Site mailing list, or if the addressee is no longer employed by your organization, please notify AFRL/IFSB, 525 Brooks Road, Rome, NY 13441-4505. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
<small>Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.</small>				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE December 1999		3. REPORT TYPE AND DATES COVERED In House
4. TITLE AND SUBTITLE X WINDOWS-BASED INTERACTIVE TEST PATTERNS FOR OVERLAID STEREOSCOPIC AND TILED DISPLAYS			5. FUNDING NUMBERS PE - 62702F PR - 558S TA - PR WU- OJ	
6. AUTHOR(S) Peter A. Jedrysik, Richard H. Sweed and *Robert VanPelt				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) AFRL/IFSB 525 Brooks Road Rome, NY 13441-4505			8. PERFORMING ORGANIZATION REPORT NUMBER AFRL-IF-RS-TR-1999-251	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFRL/IFSB 525 Brooks Road Rome, NY 13441-4505			10. SPONSORING/MONITORING AGENCY REPORT NUMBER AFRL-IF-RS-TR-1999-251	
11. SUPPLEMENTARY NOTES *Litton PRC contractor, work performed under ADII Lab Support contract. AFRL Project Engineer: Peter A. Jedrysik/IFSB/(315)330-2150.				
12a. DISTRIBUTION AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) Composite display systems consisting of multiple video projectors and/or direct view monitors are becoming more commonplace as the needs for 3D stereoscopic viewing and very high-resolution large screen displays become more prevalent. In these installations, the goal is to take separate elements and obtain a single amalgamated view. The object of this in-house task was to develop a number of interactive display patterns that would provide a more accurate, time saving means to test and align video projects in a tiled Datawall configuration and an overlaid stereoscopic configuration. Several of these composite display systems are currently implemented in the Advanced Displays and Intelligent Interfaces (ADII) Visualization Facility at the Information Directorate of AFRL, in Rome, NY.				
14. SUBJECT TERMS display test patterns, tiled display alignment, overlaid stereoscopic display alignment			15. NUMBER OF PAGES 44	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL	

Table of Contents

Table of Figures.....	iii
Abstract.....	1
1. Introduction	1
1.1 Intended Audience.....	1
1.2 Objective	1
2. Background.....	3
2.1 Overlaid Stereoscopic Display	3
2.2 Tiled Display	5
3. Related Work.....	9
3.1 GIF & JPEG Test Patterns	9
3.2 Pattern Generator Hardware.....	9
4. Interactive Test Pattern Generator Software.....	10
4.1 Silicon Graphics, Inc. Graphics Library Version.....	10
4.2 X Windows-Based Version.....	10
4.2.1 Grid.....	11
4.2.2 ZigZag	12
4.2.3 GrayScales.....	13
4.2.4 ColorScales.....	13
4.2.5 ColorWheel	14
4.2.6 WhiteBorders	14
4.2.7 VerticalLines	15
4.2.8 VerticalLadder.....	15
4.2.9 HorizontalLadder	16
4.2.10 4X128X128Blocks	16
4.2.11 LeftRightTickMarks.....	17
4.2.12 Circle/Ellipse.....	17
4.2.13 FlatField	18
4.2.14 DIDspecial.....	18
4.2.15 DotBox	19
4.2.16 GrayScalesStereo	20

4.2.17 ColorScalesStereo	21
4.2.18 HorizontalGrayScales.....	22
5. Application Execution.....	22
6. X Resource File.....	22
7. The Interfaces	23
7.1 Graphical User Interface (GUI).....	23
7.2 Keyboard	24
8. Standards-Based Software Development.....	25
9. Source Code	26
9.1 BX (Builder Xcessory) Generated	26
9.1.1 main-c.c	26
9.1.2 creation-c.c	26
9.1.3 creation-c.h	26
9.1.4 callbacks-c.c	26
9.1.5 datawall.uil	27
9.1.6 bxutils-c.c	27
9.2 Other.....	27
9.2.1 utilities.c	27
9.2.2 event_handlers.c	27
9.2.3 datawall.h	28
10. Conventions	28
10.1 Naming	28
10.2 Graphic	28
10.3 Error Handling.....	28
10.3.1 Informational.....	28
10.3.2 Fatal	29
11. Building Application	29
11.1 Makefile	29
11.2 Environment Variables.....	29
12. Adding A New Pattern.....	29

13. Summary	31
References	33

Table of Figures

Figure 1. Geometric Display Distortions	2
Figure 2a. Overlaid Configuration	4
Figure 2b. AFRL/IF's Stereoscopic 3D Display System	5
Figure 3a. 1x3 Tiled Configuration.....	5
Figure 3b. AFRL/IF's Interactive DataWall	6
Figure 4a. The Deployable Interactive DataWall (DID).....	8
Figure 4b. The DID Interior Components.....	8
Figure 5. Grid.....	11
Figure 6. ZigZag.....	12
Figure 7. 2 x 2 ZigZag.....	12
Figure 8. GrayScales	13
Figure 9. ColorScales	13
Figure 10. ColorWheel.....	14
Figure 11. WhiteBorders	14
Figure 12. VerticalLines.....	15
Figure 13. VerticalLadder	15
Figure 14. HorizontalLadder	16
Figure 15. 4X128X128Blocks.....	16
Figure 16. LeftRightTickMarks	17
Figure 17. Closeup of LeftRightTickMarks.....	17
Figure 18. Circle.....	18
Figure 19. Ellipse	18
Figure 20. FlatField.....	18
Figure 21. DIDspecial	19
Figure 22. Closeup of DIDspecial.....	19
Figure 23. DotBox.....	20
Figure 24. Close-up of Upper Left Corner of DotBox	20
Figure 25. Close-up of One Properly Aligned <i>dot-box</i>	20
Figure 26. GrayScalesStereo (Top Row Only)	21
Figure 27. GrayScalesStereo (Both Rows)	21
Figure 28. ColorScalesStereo (Top Row Only)	21
Figure 29. ColorScalesStereo (Both Rows)	21
Figure 30. HorizontalGrayScales	22

Abstract

Composite display systems consisting of multiple video projectors and/or direct view monitors are becoming more commonplace as the needs for 3D stereoscopic viewing and very high-resolution large screen displays become more prevalent. In these installations, the goal is to take separate elements and obtain a single amalgamated view. The objective of this in-house task was to develop a number of interactive display patterns that would provide a more accurate, timesaving means to test and align video projectors in a tiled DataWall configuration and an overlaid stereoscopic configuration. Several of these composite display systems are currently implemented in the Advanced Displays and Intelligent Interfaces (ADII) visualization facility at the Information Directorate of the Air Force Research Laboratory (AFRL/IF) in Rome, New York.

1. Introduction

1.1 Intended Audience

This report introduces the reader to basic concepts of high resolution, projected displays, the problems associated with such displays, and how this application can help solve the problems. It is designed to aid anyone using or maintaining this test pattern generation software including information required for compiling and linking this application. It is assumed that the user has a basic knowledge of UNIX, X Windows, X library (i.e. Xlib) graphics, Motif and the Integrated Computer Solutions' (ICS) Builder Xcessory (BX) [3] graphical user interface (GUI) development tool.

1.2 Objective

The original objective of this in-house task was to develop a Sun Microsystems-based version of a test pattern generation application previously developed for the Silicon Graphics, Inc. (SGI) workstations that used the SGI graphics library (GL) [4]. After a short investigation of potential solutions, we decided to create a more universal application based upon X Windows rather than simply port the application to the Sun platform. The SGI-based application is capable of generating a number of test patterns used to align and adjust various characteristics of high resolution, tiled or overlaid stereoscopic projected displays. The X Windows-based application provides a superset of the test patterns provided by the SGI-based application. The new application uses the X primitives to generate all graphics rather than the SGI GL functions. X Windows was chosen as the development standard due to portability, availability and economic considerations. Using this application greatly improves composite display image quality by providing a more accurate and less time consuming method for registration, color balance and alignment of the video projectors.

Several interactive display test patterns were designed and implemented that provide a comprehensive testing and alignment environment for a uniform seamlessly tiled $n \times m$ display or an overlaid stereoscopic display. The issues that needed to be addressed included: (1) ensure each projector is displaying all pixels around its border, (2) allow horizontal alignment of side by side tiled displays and include a capability to provide vertical alignment for future tiled displays greater than $1 \times n$, (3) ensure accurate color balance among all projectors to assure continuous color balance across the display, (4) minimize geometric display distortions such as pincushioning, keystoneing, and vertical/horizontal nonlinearity (Fig. 1), (5) test the capacity of the display system to resolve very high-resolution images, and (6) allow overlaid alignment of a large screen stereoscopic display.

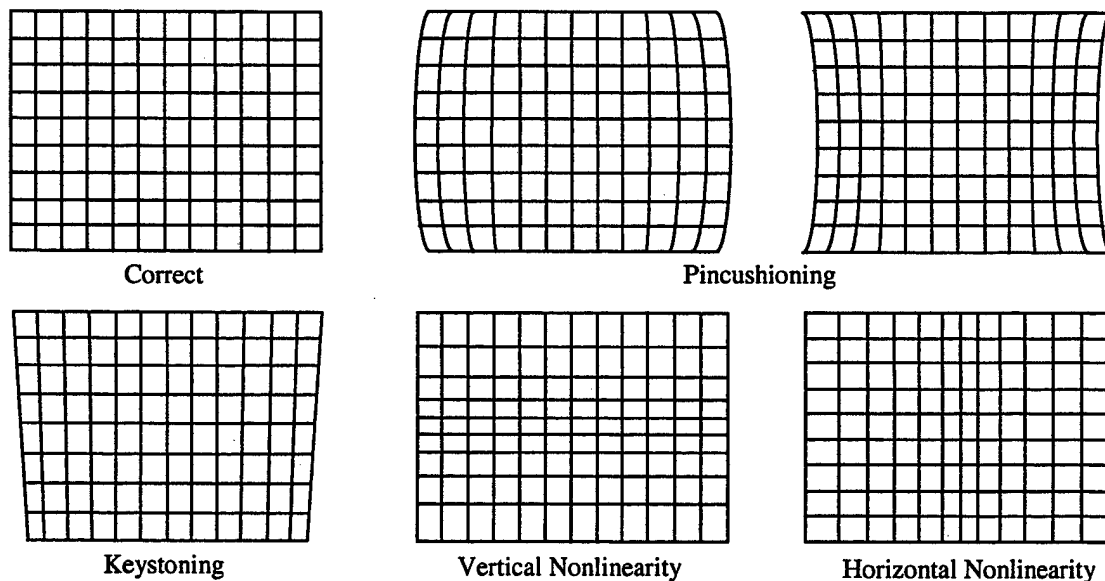


Figure 1. Geometric Display Distortions

All the video projectors used in the ADII visualization facility are either cathode ray tube (CRT) based systems composed of separate red, green, and blue CRTs, or liquid crystal display (LCD) based systems. A key advantage of the CRT based display systems is the ability to correct display image geometry. Display distortions can be isolated to specific components and adjusted in a number of ways both mechanically and electronically. Individual CRTs and isolated portions of the display image can be adjusted independently. It does, however require a significant amount of effort to optimize the display image. Most LCD projectors have the distinct advantages of being much brighter, smaller, lighter, and easier to set-up. Their shortfalls, however, include lower resolution, and the inability to correct display distortions through geometry adjustments. The latter necessitates precision optics to minimize these distortions, particularly in a tiled configuration.

All CRT video projectors have some type of internal test pattern generator that is built into the projectors by the manufacturer to aid in focusing and converging the CRTs. It provides a step-

by-step process that includes manual focus and electronic adjustment of the CRTs to correct geometric display distortions. The only patterns they provide, however, are typically a grid or a cross hair with a fixed line width. None of the LCD projectors in use at the ADII facility provide any such internal test pattern generation. Although the CRT projectors' built-in patterns can be used for some of the preliminary tuning, they were never intended to address the many issues associated with our very unique multiple projector display configurations. We have very specific needs with regard to alignment and color balance. A conventional large screen display would typically only consist of a single projector, where exact image positioning on the projection screen and color balance with other projectors isn't required. Alignment and color balance are critical for tiled configurations to reduce the distraction of the seams and provide a continuous display image. They are essential in overlaid stereoscopic displays to ensure the left and right eye images can be fused effectively.

An important aspect of the test pattern environment that was developed is that rather than using any internal test pattern capability, the projectors are being driven by the computer system, which also generates the display imagery in our working environments. This provides a more accurate representation of a working environment display, and allows for a more accurate test of the entire display system. It takes into account all external and internal distortions; problems introduced by the computer, communication cabling, distribution hardware, and the projectors. This ensures more accurate tuning of the displays to produce better and more consistent images than would otherwise be provided using internally generated patterns.

2. Background

2.1 Overlaid Stereoscopic Display

An overlaid stereoscopic display configuration consists of dual video projectors each rear projecting either left or right eye images overlaid across the same screen area (Fig. 2). In rendering a 3D scene on the computer, the location of the "eye" is defined as one of the parameters that determines the viewing volume, and subsequently which objects within the scene will be visible, i.e. within the field of view. To mimic the way the eyes view 3-dimensional objects in the real world, the left and right eye scene renderings have eye positions with a lateral disparity equal to an average interpupillary distance (distance between the eyes) for the viewer. The images are kept separate and distinct by passing each projected image through a polarizing filter (one polarized orthogonal to the other). A viewer observes the images through glasses, similarly polarized. Left and right eye information is only seen by each appropriate eye to create the illusion of depth.

Unfortunately the process of polarization is very lossy with each pass through a polarizing filter transmitting only 25% of the incident light. Since the process involves two passes through the filter material for each image, only a small percentage of the original light is available to the viewer.

To compensate for the light loss and to preserve the polarization of the incident light on the screen, a special screen construction technique is used in the 3D stereoscopic display installation at AFRL/IF. The substance of the screen is an acrylic panel, 88" x 66" x 1/4" thick. The rear projection surface is etched with a Fresnel surface that creates an image plane and at the same time preserves the polarization of the incident light. The front surface of the screen contains a lenticular lens pattern, providing an apparent screen gain of 5, which offsets a portion of the polarization losses. One artifact of this specialized screen construction is that, because the image plane and lenticular surface are not coincident, the absolute resolving power of the screen is limited. In this particular instance, pixels of 1/16" and larger are individually resolvable.

In the original installation, the images were provided by a pair of CRT projectors whose projected resolution was 1280 x 1024 pixels with a light output of less than 100 lumens. In order to maintain a minimally usable screen illumination of approximately 2-3 foot-lamberts (roughly the brightness of a drive-in movie image), the maximum image size was limited to 40" x 30". As the images passed through the screen with its 1/16" pixel resolution, overall resolution was reduced to 640 x 480 pixels.

The current installation has replaced the CRT projectors with LCD-based units that resolve 1024 x 768 pixels with a light output of at least 1200 lumens. The increased intensity permitted extending the image to the full 88" x 66" dimensions of the screen while improving the apparent screen illumination to an estimated 16 foot-lamberts. The 88" wide image produces a pixel size of .085" which is not noticeably affected by the 1/16" resolving capability of the screen.

The current installation, therefore, improves image area by a factor of 4 + (40" x 30" to 88" x 66"), improves image brightness by a factor of 8 (2 to 16 foot-lamberts), and improves resolution from 640 x 480 to 1024 x 768 pixels.

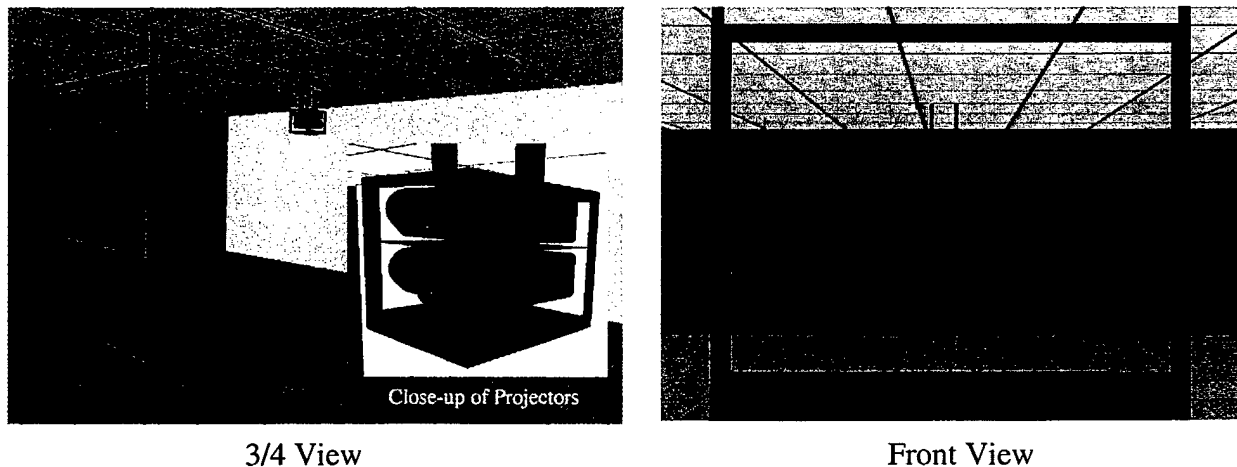


Figure 2a. Overlaid Configuration

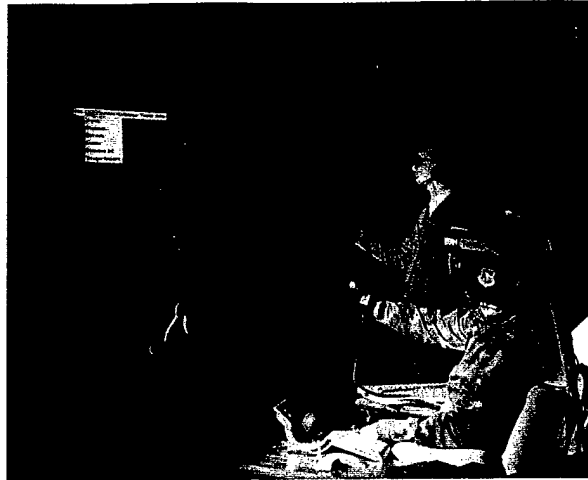
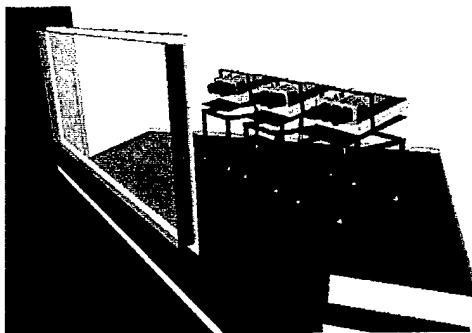


Figure 2b. AFRL/IF's Stereoscopic 3D Display System

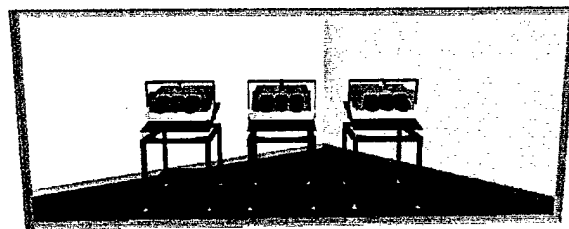
Primary concerns with implementing an overlaid stereoscopic display are assuring accurate color balance, vertical and horizontal linearity, and proper alignment of the left and right eye displays. This ensures effective fusing of the left and right eye stereo pairs into what your brain will interpret as a single, coherent, 3-dimensional image.

2.2 Tiled Display

A tiled display consists of $n \times m$ distinct display devices, such as video projectors, each displaying a portion of an entire screen area (Fig. 3). The intent is to produce a very high-resolution, seamless, large screen display. Again accurate color balance and proper alignment of the display devices are crucial. Variations in chromaticity and luminosity among tiles can cause inconsistency in color and brightness across the screen. Vertical or horizontal disparity of the tiled images can cause segmentation of objects at the seams. Gaps between image tiles can cause discontinuity of the imagery.



3/4 View



Front View

Figure 3a. 1 x 3 Tiled Configuration



Figure 3b. AFRL/IF's Interactive DataWall

The rationale for this type of display is fourfold. First, a *large screen display* provides multiple users with a common display medium, coordinating data from multiple workstations and monitors. It provides a large canvas to present multiple windows with various information. Windows can be spread out instead of continuously bringing the active window to the foreground. It provides a global view of the information space. This allows the users to make comparisons and find relationships between items. It also makes collaboration within a localized working environment much more effective.

Second, *high-resolution* is desired because AFRL/IF is researching the Interactive DataWall for military Command and Control, which requires the display of various kinds of data. This data can include terrain models overlaid with computer-generated imagery, digital maps, textual information, as well as live and recorded video. Although a conventional projection system allows a wide range of image sizes, and could provide the necessary large screen display capability, the display quality will inevitably decrease as the display area increases and pixels become larger. The DataWall is intended to be used both at a distance and at close range. A large screen display's utility is significantly reduced if imagery loses important detail or text becomes difficult to read at close proximity. Even today, large paper maps with acetate overlays are used for mission planning in the command centers. Therefore, near paper map quality is required to effectively replace conventional data sources with electronic media.

Third, *tiling* several images either horizontally and/or vertically provides a wider field of view, and allows displays of unlimited aspect ratios (i.e. the height to width ratio) to be created. Simply increasing a projector's display area will introduce pixelation problems, and will decrease the brightness of the image. In a tiled configuration each projector is displaying a small portion of the entire display area. The combined image is much brighter than a single projector displaying an image across the same screen area. By limiting each projector's display area, the projected

light is more concentrated. Direct view displays are much brighter and less susceptible to ambient light problems than projection systems. However, implementing a single element direct view display creates a myriad of new problems. Scaling already bulky CRTs or flat panels is neither practical nor economical. Scaling up CRTs beyond a 40" diagonal would require glass envelopes that are heavy and high voltages that are radiation hazards. Fabrication of very large-scale flat panels would require the development of very expensive equipment [1]. Larger flat panel displays are becoming available, but are far from the resolution capacity of the CRT systems. In addition, the state-of-the-art in display technology is limited to devices that are only capable of resolutions on the order of 2500 x 2000 pixels. Tiling the highest resolution display devices in an $n \times m$ configuration increases the display resolution $n \times m$ -fold.

Last, *projectors* were utilized to minimize the seams between the image tiles. A common implementation for tiled displays is to use direct view monitors. All currently available direct view monitors have frames that enclose a certain amount of necessary electronics surrounding the display screen. Therefore, there is no way to effectively abut direct view display elements without having a gap between them.

AFRL/IF has successfully implemented a number of DataWalls each consisting of three horizontally tiled video projectors. The first implementation, which serves as a development system, uses CRT projectors each displaying 1600 x 1200 pixels for a total display resolution of 4800 x 1200 pixels across a screen area 12' x 3' (Fig. 3). This far exceeds the state-of-the-art in single element display systems. An SGI Onyx workstation with three Reality Engines drives the display. In addition, each projector has a video bandwidth capacity approaching 2500 x 2000 pixels, which could yield a future DataWall resolution of 15 million pixels.

In the interest of developing a less costly alternative to the SGI-based DataWall, a Sun workstation based version was successfully implemented. Again three CRT projectors were utilized but at a slightly reduced resolution. Each projector displays 1280 x 1024 pixels for a total display resolution of 3840 x 1024 pixels across a screen area 12' x 3'. One dual processor Sun Ultra SPARC 60 workstation and two Ultra SPARC 30 workstations drive the display.

To provide a deployable version of the Interactive DataWall to support the testbed for the forward deployable element of the Configurable Aerospace Command and Control (CACC) Integrated Technology Thrust Program (ITTP), a Sun workstation based Deployable Interactive DataWall (DID) was implemented. One dual processor Sun Ultra SPARC 60 and two Ultra SPARC 10s drive this display. It is housed in an extensively modified Air Force S-530 A/G Standard Rigid Walled shelter, with its own Tactical Generator Set and Environmental Control Unit. Due to the unique, short-throw, rear-projection requirements, three LCD projectors with special short-throw lenses are used. In this configuration, each projector displays 1024 x 768 pixels for a total display resolution of 3072 x 768 pixels across a screen area 9' x 2 1/4' (Fig. 4). It should be noted that even in this most reduced resolution configuration, the total display resolution still exceeds the state-of-the-art in single element display systems.

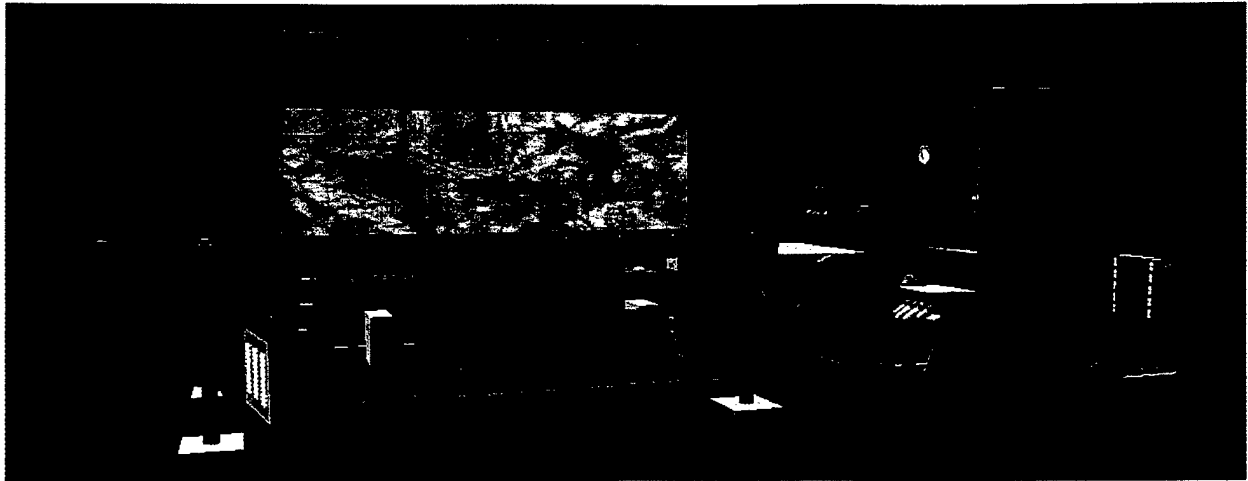


Figure 4a. The Deployable Interactive DataWall (DID)

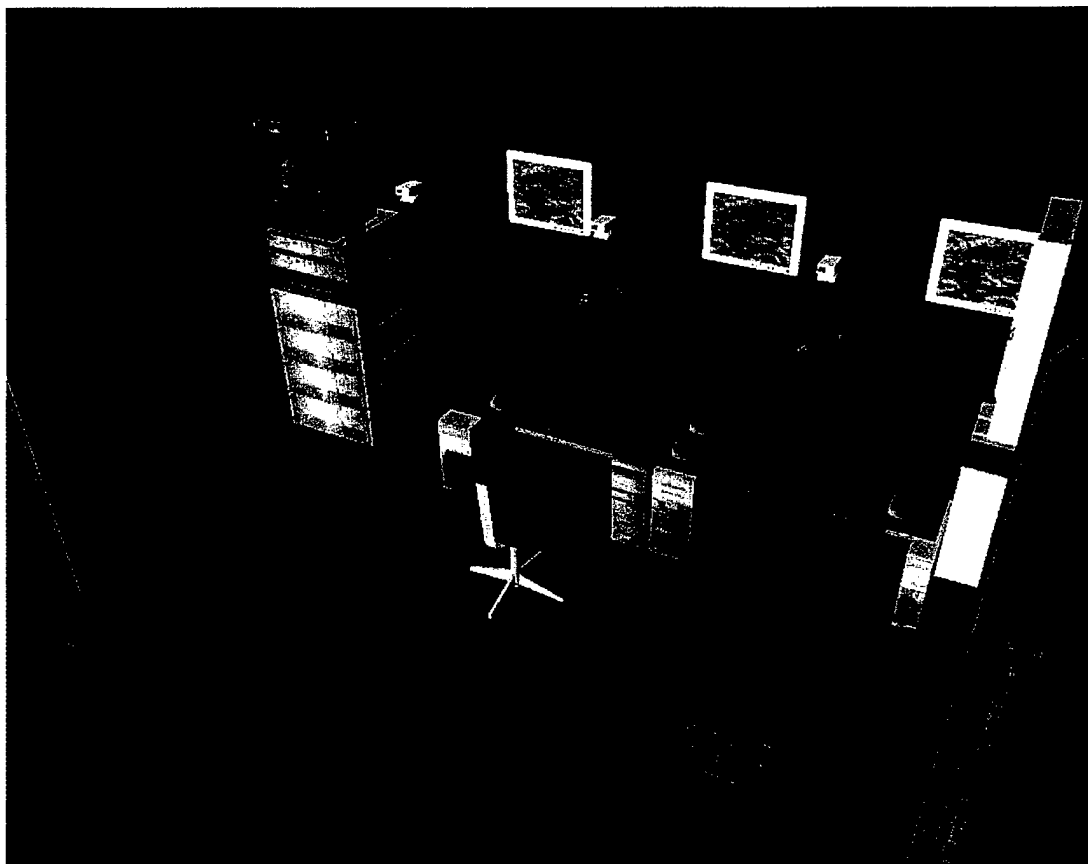


Figure 4b. The DID Interior Components

A significant challenge of producing a seamless display is ensuring the projectors are properly aligned with correctly balanced color and brightness across the individual projector images. It should resemble a monolithic display and look like a single continuous image; not a mosaic of varying colors and brightness caused by chromaticity and luminosity variations. Nor should there be segmented lines or gaps between the tiles caused by inaccurate alignment.

3. Related Work

Overlaid and tiled display systems have been in use, albeit limited, for the past several years. However, little has been done to develop test patterns for these special display configurations. Test patterns for determining display system image quality have been developed and used for the past several years. Some of the display metrics they were designed to test includes linearity, color, sharpness/resolution, contrast, video artifacts, and focus. They were not designed with composite display requirements in mind, however, and are better suited to single element display systems.

3.1 GIF & JPEG Test Patterns

In addition to the internally generated test pattern mechanisms found in commercial display systems described earlier, test patterns, typically in Graphical Image Format (GIF) and Joint Photographic Experts Group (JPEG) formats are also available. What they provide are static images such as grid patterns, line patterns, or color bars. Although most could be used in a composite display system, there are more effective images that could be used in the alignment process. In the case of a tiled display they do not provide the necessary imagery along the seams for color balancing and alignment. Their greatest shortfall is the lack of interactivity. In a composite display configuration, the ability to change image characteristics such as colors and line widths provides significant improvement over the static patterns.

3.2 Pattern Generator Hardware

Special hardware is also available that can be installed as an add-on board to your image generation computer hardware that provides video test signal/pattern generation. They provide similar functionality to the GIF and JPEG test patterns, in that they are static patterns designed to test and measure the performance of video production equipment. Typical uses include video signal quality control and management, video inspection and testing, maintenance/calibration, and troubleshooting & repair. The same deficiencies apply with these types of test environments. They do not allow interactive changes to the images and are not really designed for aligning multiple display elements.

4. Interactive Test Pattern Generator Software

4.1 Silicon Graphics, Inc. Graphics Library Version

The first version of the interactive test pattern generation application was designed for the SGI workstations that were being utilized to drive both the DataWall and overlaid stereoscopic displays in the ADII visualization facility. Seven patterns were developed to address the above issues, and are all accessible through an interactive C program that implements Graphics Library (GL) function calls to render each screen pattern. Each pattern can be displayed on systems of varying resolutions. The resolution is not something that needs to be explicitly specified for any particular display; it is a screen characteristic that is obtained through a GL function call internal to this application. There are no minimum or maximum resolution limitations to the application [4].

The first implementation of the DataWall at AFRL/IF prompted the need for an alignment and color balancing test environment. The test patterns provided a significantly improved method for aligning the three projectors and effectively balancing the color across the entire display area. This tool has been evolving as the DataWall evolves, with improvements made as additional needs were identified. With the original DataWall implementation being SGI-based, the GL programming approach to the test pattern generator development served the system well.

4.2 X Windows-Based Version

The migration of the DataWall to a Sun-based implementation drove the requirement for a more generic test pattern generator. GL is a set of graphics libraries specific to the SGI computer platform. The X Windows-based application provides a superset of the test patterns provided by the SGI-based application. The new application uses the X primitives to generate all graphics rather than the SGI GL functions. This application generates the following patterns: Grid, ZigZag, GrayScales, ColorScales, WhiteBorders, VerticalLines, ColorWheel, VerticalLadder, HorizontalLadder, 4X128X128Blocks, LeftRightTickMarks, Circle, Ellipse and FlatField. To generate these patterns, the application uses twelve colors (Table 1) and nineteen grayscale levels. Each color component is 16 bits and intensities are expressed as percentages of the maximum value for an unsigned 16 bit integer (i.e. 65536). The grayscales used in this application are selected linearly from the range between black (0% red, 0% green, 0% blue) and white (100% red, 100% green, 100% blue). Grayscales have the same intensity value for each of the three color components (i.e. red, green & blue). Collectively, these patterns are intended to aid the video engineer in focusing, aligning, color balancing and performing other adjustments to high resolution, projected displays.

X Windows Color Name	Red Intensity	Green Intensity	Blue Intensity
Red (red)	100%	0%	0%
OrangeRed (orange red)	100%	0%	50%
Magenta (magenta)	100%	0%	100%
MediumSlateBlue (medium slate blue)	50%	0%	100%
Blue (blue)	0%	0%	100%
SlateBlue (slate blue)	0%	50%	100%
Cyan (cyan)	0%	100%	100%
SpringGreen (spring green)	0%	100%	50%
Green (green)	0%	100%	0%
Chartreuse (chartreuse)	50%	100%	0%
Yellow (yellow)	100%	100%	0%
Coral (coral)	100%	50%	0%

Table 1. Pattern Colors

Note: A color name enclosed in ()'s is simply an alternate name for the same color.

4.2.1 Grid

The Grid pattern is the default pattern. It is the initial pattern displayed when the application executes. The pattern consists of 21 evenly distributed vertical white lines and 17 evenly distributed horizontal white lines on a black background. The width of the lines is variable from one to ten pixels (Fig. 5). This pattern was designed to determine the level of pincushioning, keystoneing, and horizontal or vertical linearity distortions present in a displayed image (Fig. 1). The squares that compose the grids should appear square. If the outside edges of the image are sloped and/or curved, there are keystoneing and/or pincushioning distortions. If the squares appear to become more rectangular towards the center or the outside of the image, either horizontally or vertically, there is a linearity problem. This pattern can be used for aligning tiled as well as overlaid stereoscopic displays. Although this pattern can be used effectively to adjust the types of distortions described above, it can overwhelm the eye with information and, as a consequence, can be difficult to use. It is easier to make the linearity adjustments using the VerticalLadder and HorizontalLadder patterns, which will be discussed later, and use the Grid pattern to confirm the settings.

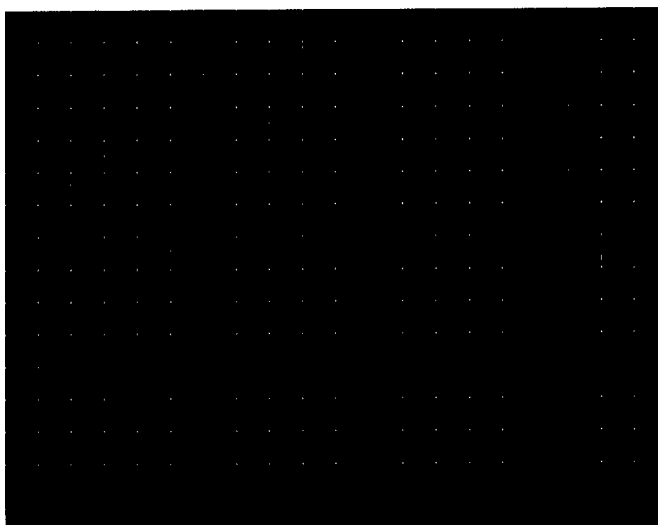


Figure 5. Grid

4.2.2 ZigZag



The ZigZag pattern assists in horizontally aligning a tiled display, but was also designed to handle alignment of future vertically tiled displays. The pattern consists of a black background with edge zigzag patterns (Fig. 6). When these patterns are abutted in a tiled configuration, they produce a continuous zigzag pattern across adjacent screens along the seams (Fig. 7). Each zigzag consists of seven adjacent lines, each one pixel wide and one of the following colors: red, green, blue, cyan, yellow, magenta and white. The lines can vary in width from one to ten pixels.

Figure 6. ZigZag

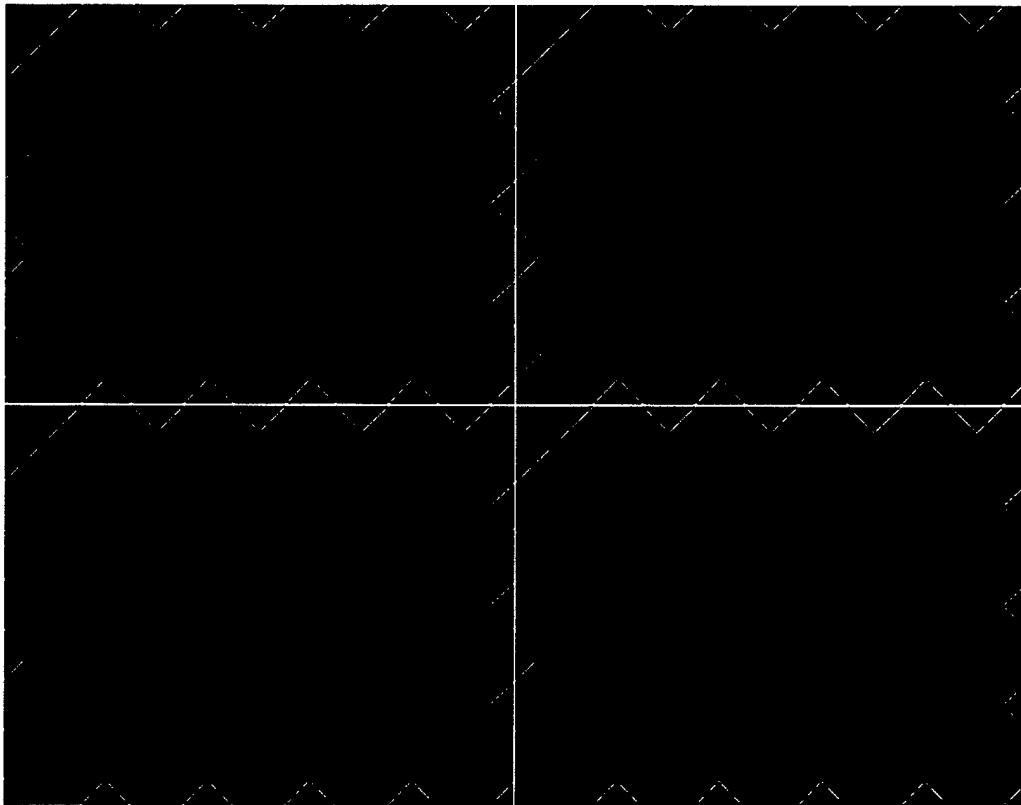


Figure 7. 2 x 2 ZigZag

Note: The seams depicted serve only to delineate each projector's screen area. The goal of such a tiled display is to avoid any seams.

4.2.3 GrayScales

The GrayScales pattern is two rows of 16 equal-sized, adjacent, vertical rectangular areas. Each rectangle is flooded with one of the grayscale levels available to the application. The sole difference between the top and bottom rows is a reversal of the grayscale level presentation order (Fig. 8). Because the full spectrum of grayscales from white to black consists of equal amounts of red, green and blue, it was determined that a pattern consisting of grayscale bands could provide an effective color-balancing tool.

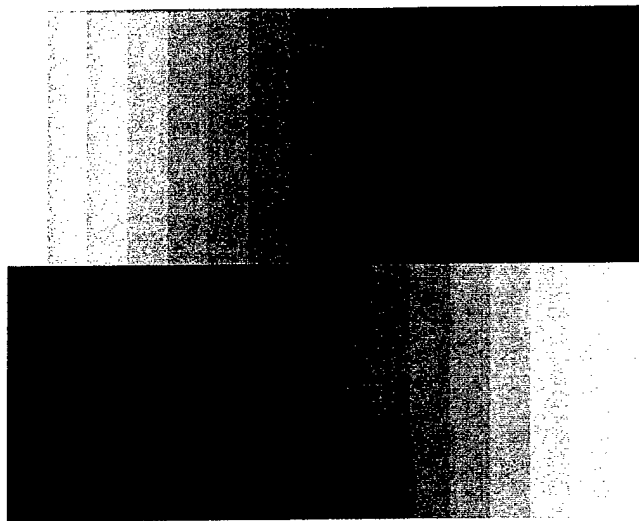


Figure 8. GrayScales

4.2.4 ColorScales

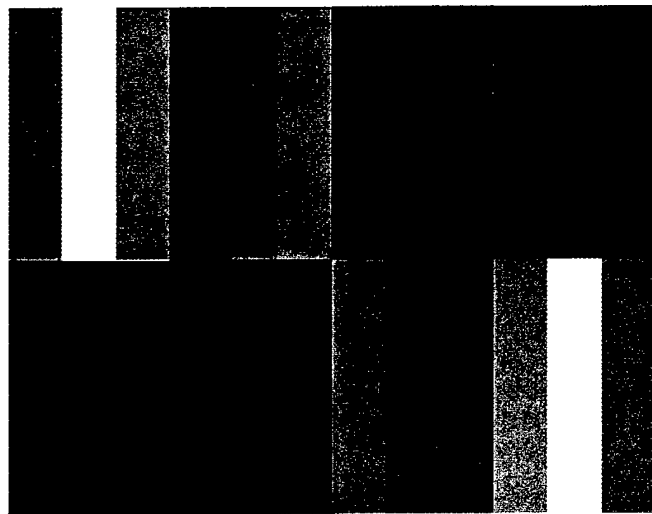


Figure 9. ColorScales

The ColorScales pattern is two rows of 12 equal-sized, adjacent, vertical rectangular areas. Each rectangle is flooded with one of the standard colors available to the application. The sole difference between the top and bottom rows is a reversal of the color presentation order. The colors used are red, orange red, magenta, medium slate blue, blue, slate blue, cyan, spring green, green, chartreuse, yellow and coral (Fig. 9). This pattern is used for color balancing.

4.2.5 ColorWheel

The ColorWheel pattern contains 12 polygons flooded with one of the colors available via the internal application colormap. At the center of the color wheel is a white circle. The colors used are red, orange red, magenta, medium slate blue, blue, slate blue, cyan, spring green, green, chartreuse, yellow and coral (Fig. 10). This pattern is used for color balancing and can be rotated to align and compare colors along adjacent tile seams.

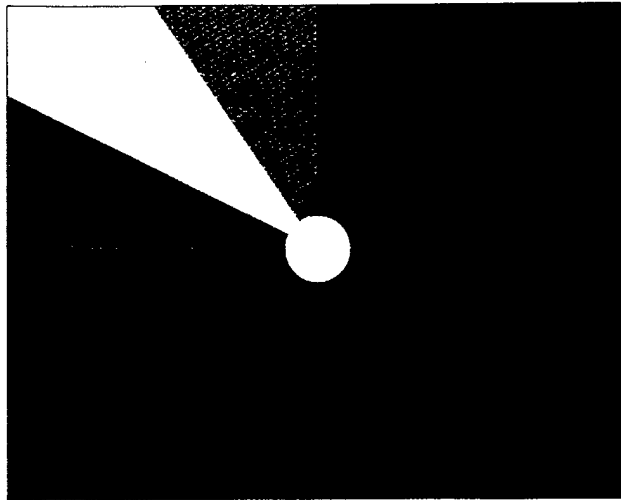


Figure 10. ColorWheel

4.2.6 WhiteBorders

The WhiteBorders pattern is a series of eight concentric rectangles. All rectangles are displayed using 'n' pixel wide white lines on a black background. The line width is variable from one to ten pixels. The outermost rectangle has a height equal to the vertical resolution and a width equal to the horizontal resolution of the display. The height and width of each subsequent rectangle equals the height and width of the previous rectangle reduced by four times the line width. The origin of the outermost rectangle is the pixel in the top right corner (i.e. row = 0 and column = 0). The origin of subsequent rectangles is offset in both the x and y dimensions by two times the line width (Fig. 11). This pattern was designed to ensure each projector is displaying all pixels around its border and is useful for evaluating resolution.

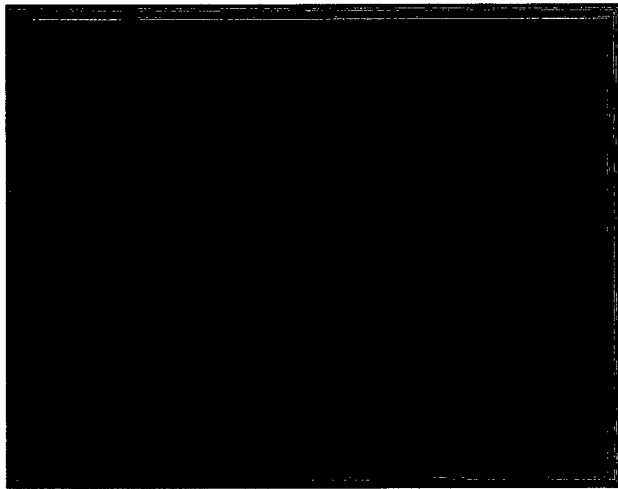


Figure 11. WhiteBorders

4.2.7 VerticalLines

The VerticalLines pattern is a series of vertical lines filling the display area. All lines are white on a black background. Each line is n pixels wide and is spaced n pixels from adjacent lines. The line width and spacing is variable from one to ten pixels (Fig. 12). The intent of this pattern is to test the capacity of the display system to resolve very high-resolution images.

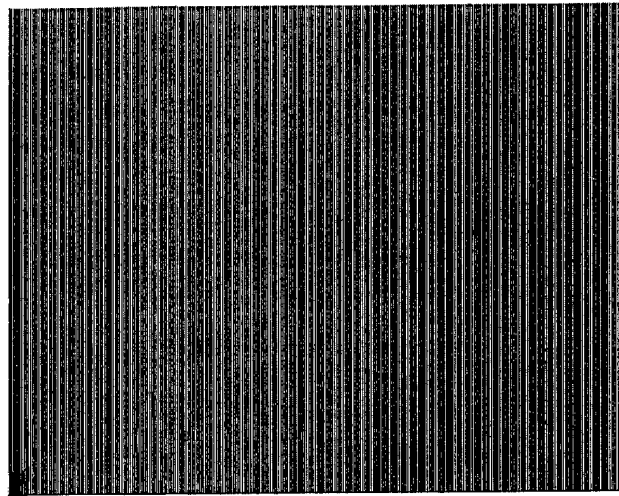


Figure 12. VerticalLines

4.2.8 VerticalLadder

The VerticalLadder pattern consists of a rectangular border with a horizontal line connecting the midpoints of the two sides of the rectangle and a centered, vertical *ladder*. The ladder portion of the pattern is comprised of two parallel, vertical lines connecting the top and bottom rectangle borders with endpoints spaced equidistant from the midpoint of each border (Fig. 13). Initially, this pattern is generated with red lines on a black background but the foreground color (i.e. lines) can easily be modified via a foreground color selection window (refer to Section 7.1). This pattern is used for adjusting vertical linearity. It is much easier to use than the Grid pattern since it provides required information in a much less distracting manner.



Figure 13. VerticalLadder

4.2.9 HorizontalLadder

The HorizontalLadder pattern is a rectangle with a vertical line connecting the midpoints of the top and bottom of the rectangle and a centered, horizontal *ladder*. The ladder portion of the pattern is comprised of two parallel, horizontal lines connecting the left and right rectangle borders with endpoints spaced equidistant from the midpoint of each border (Fig. 14). Initially, this pattern is generated with red lines on a black background but the foreground color (i.e. lines) can easily be modified via a foreground color selection window (refer to Section 7.1). This pattern is used for adjusting horizontal linearity. It is much easier to use than the Grid since it provides required information in a much less distracting manner.



Figure 14. HorizontalLadder

4.2.10 4X128X128Blocks

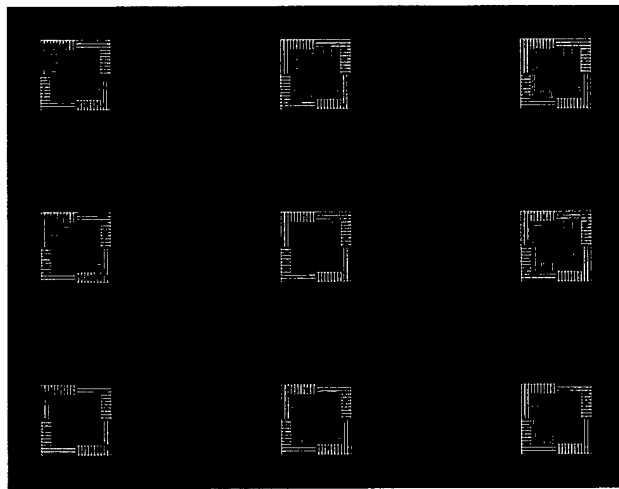


Figure 15. 4X128X128Blocks

The 4X128X128Blocks pattern is a compound pattern made up of nine 128 pixel by 128 pixel blocks distributed on the display area. The nine blocks are displayed in three rows of three blocks. Each of these blocks is made up of four 64 pixel by 64 pixel sub-blocks. Each sub-block is made up of a pattern of varying width (one to three pixels) horizontal and vertical white lines on a black background. Starting with the upper left sub-block and moving clockwise, the pattern is rotated 90 degrees for each subsequent sub-block (Fig. 15). This pattern is used primarily to aid focus adjustments.

4.2.11 LeftRightTickMarks

The LeftRightTickMarks pattern is a rectangle with alternating short and long tick marks starting at the left and right borders and extending towards the middle of the rectangle. Short tick marks are 64 pixels in length while long tick marks are 128 pixels in length. There are 15 tick marks (eight short and seven long) along each border. The topmost tick mark is short and they alternate from long to short until all 15 tick marks are displayed (Fig. 16). The line forming the rectangular border starting at a short tick mark and extending to the next long tick mark is two pixels wide, otherwise, the rectangular border is one pixel wide (Fig 17). Alternating the border width aids in adjusting the position of adjacent tiled projections to one pixel precision.

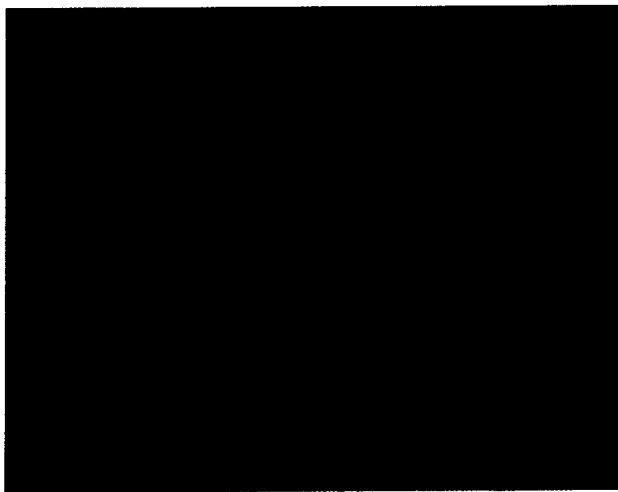


Figure 16. LeftRightTickMarks



Figure 17. Close-up of LeftRightTickMarks

Note: 1 pixel indentation along border

4.2.12 Circle/Ellipse

The Circle/Ellipse pattern is a window bordered by a white rectangle with a white circle or ellipse centered within the rectangle (Fig. 18 & Fig. 19). This pattern is used as a test to determine whether the display uses a square or rectangular pixel. This pattern is most useful for post adjustment distortion detection since the human eye is very sensitive to spherical distortions.

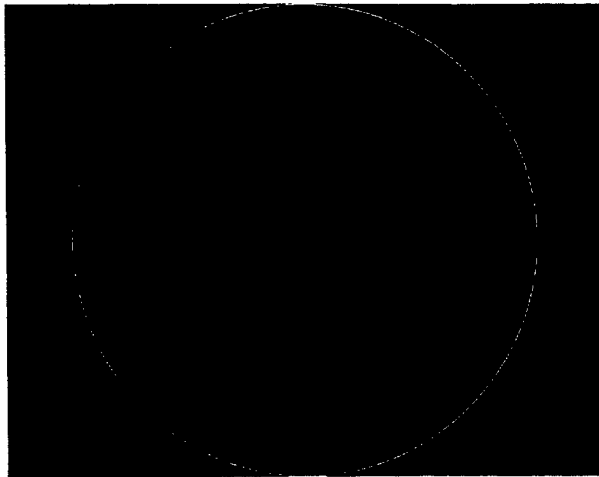


Figure 18. Circle

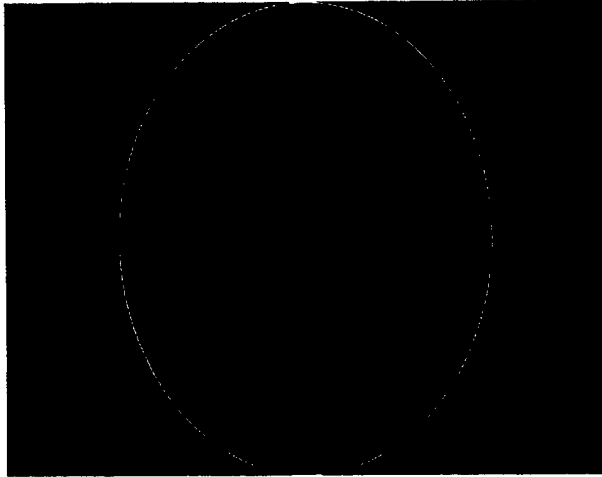


Figure 19. Ellipse

4.2.13 FlatField

The FlatField pattern floods the entire drawing area with a single color (Fig. 20). Initially, the color is an 18% gray but it is alterable via a pop-up window (refer to Section 7.1) containing a slider for each RGB component. This pattern is used to aid color balancing. It is also useful for adjusting dynamic brightness on projectors supporting such adjustments.



Figure 20. FlatField

4.2.14 DIDspecial

The DIDspecial pattern is a red rectangle with a horizontal line and a vertical line forming a perpendicular intersection at the center, and alternating short and long tick marks. The tick marks start at the left and right borders and extend towards the middle of the rectangle. Short tick marks are 64 pixels in length while long tick marks are 128 pixels in length (Fig. 21). The line forming the rectangular border starting at a short tick mark and extending to the next long tick mark is two pixels wide, otherwise, the rectangular border is one pixel wide (Fig. 22). Alternating the border width aids in adjusting the position of adjacent tiled projections to one pixel precision.

This pattern is similar to the LeftRightTickMarks pattern, but was designed for the LCD projectors used in the Deployable Interactive DataWall (DID) configuration. Because the LCD projectors are lower resolution the image consists of fewer tick marks. Also, since these projectors do not have electronic geometry adjustments, the center vertical line is used to determine true vertical, as the physical pitch of each projector is adjusted. The center horizontal line is then used to determine true horizontal, as the physical yaw of each projector is adjusted. The horizontal line is also used along with the tick marks to properly align all the projectors along the horizontal.



Figure 21. DIDspecial



Figure 22. Close-up of DIDspecial
Note: 1 pixel indentation along border

4.2.15 DotBox

The DotBox pattern consists of two distinct images that are designed to be superimposed across the same screen area in an overlaid configuration. One image has a red border and five rows of five equally spaced green dots. Each dot is two pixels by two pixels. The other image has a green border and five rows of five equally spaced red boxes. Each box is six pixels by six pixels. On an overlaid projection system, one projector displays the dot image while the other projector displays the box image (Fig. 23-25). This pattern is designed to aid in adjusting the alignment of overlaid projections by centering the dots inside the boxes, and matching the image borders.

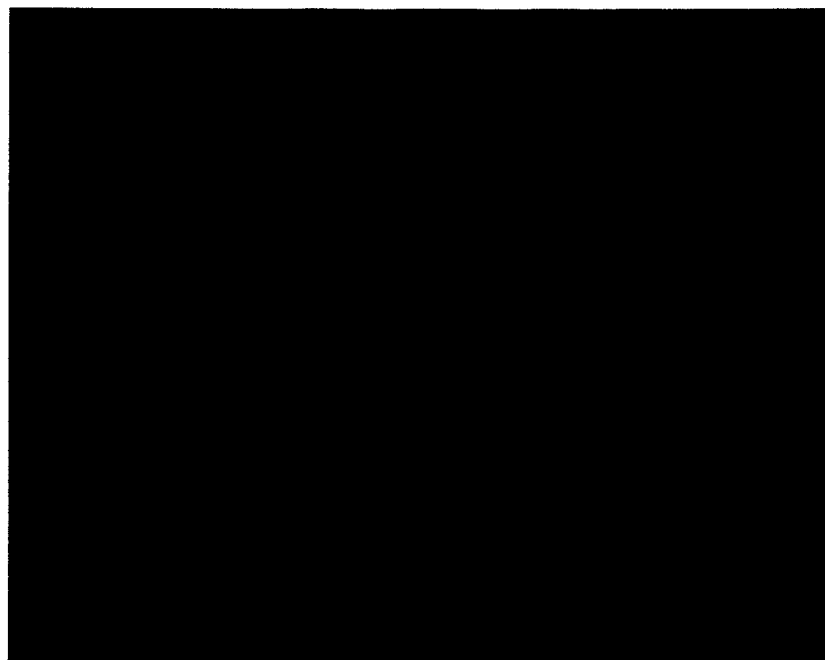


Figure 23. DotBox

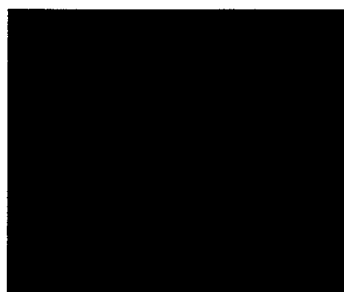


Figure 24. Close-up of Upper Left Corner of DotBox
Note: Yellow border from red and green borders being overlaid



Figure 25. Close-up of One Properly Aligned *dot-box*

4.2.16 GrayScalesStereo

The GrayScalesStereo pattern consists of two rows of 32 equal-sized, adjacent, vertical rectangular areas. Used in an overlaid configuration, one projector displays the top row, while the other projector displays the bottom row. Starting from the rightmost rectangle, each rectangle is flooded with one of 16 grayscale levels from black to white, then in reverse order from white to black. The main menu (refer to Section 7.1) contains a *buffer* submenu for displaying the top row only (Fig. 26), the bottom row only, or both rows of this pattern (Fig. 27). This pattern is designed to aid gamma matching of different projectors in an overlaid configuration.

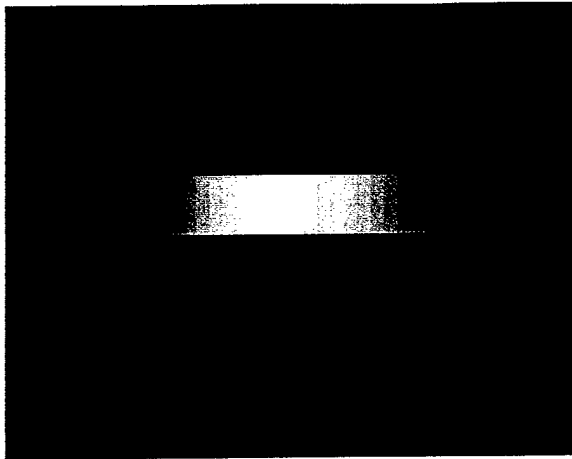


Figure 26. GrayScalesStereo (Top Row Only)

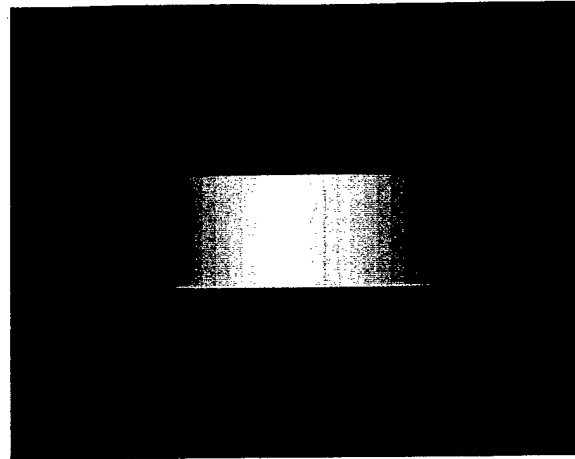


Figure 27. GrayScalesStereo (Both Rows)

4.2.17 ColorScalesStereo

The ColorScalesStereo pattern consists of two rows of 24 equal-sized, adjacent, vertical rectangular areas. In an overlaid configuration, one projector displays the top row, while the other projector displays the bottom row. Starting from the rightmost rectangle, each rectangle is flooded with one of 12 colors available to the application, then the same 12 colors in reverse order. The main menu (refer to Section 7.1) contains a *buffer* submenu for displaying the top row (Fig. 28), the bottom row, or both rows of this pattern (Fig. 29). This pattern is designed to aid color matching of different projectors in an overlaid configuration.

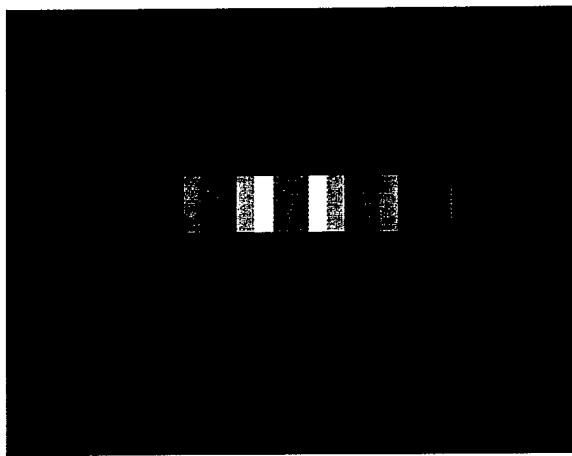


Figure 28. ColorScalesStereo (Top Row Only)

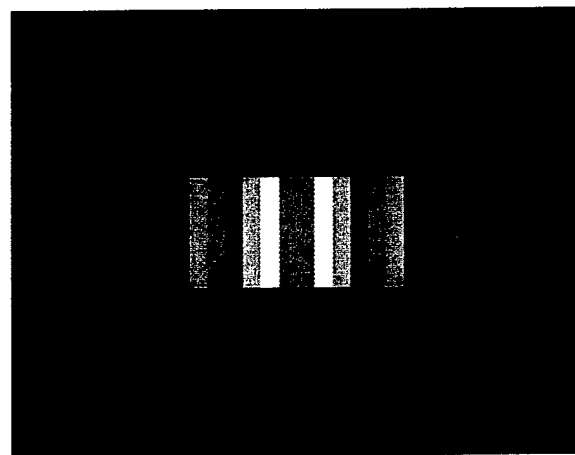


Figure 29. ColorScalesStereo (Both Rows)

4.2.18 HorizontalGrayScales

The HorizontalGrayScales pattern consists of nine horizontal, rectangular areas separated by black rectangular buffer zones. The topmost rectangle is flooded with white, followed by 20% gray, 40% gray, 60% gray, white, 60% gray, 40% gray, 20% gray and white (Fig. 30). This pattern is designed to aid gamma matching of adjacent, horizontally tiled projections.

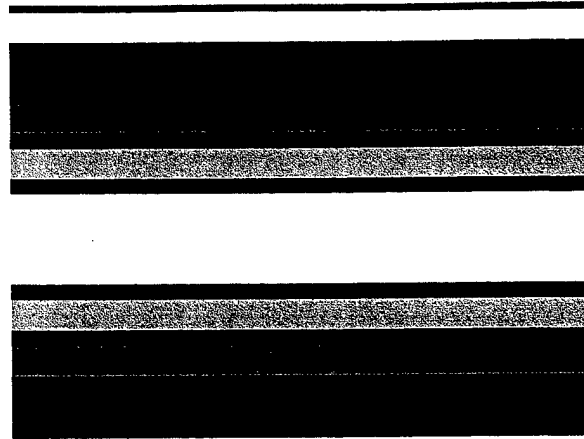


Figure 30. HorizontalGrayScales

5. Application Execution

The remaining sections describe how to execute the application, use the interfaces to interact with the patterns, recompile the application if changes are required, and add new patterns. It also describes how the source code was developed.

The test pattern generation executable (i.e. binary) is named *main-c*. To execute this application, the user should set the current working directory to the directory where the executable and an X resources definition file named *Main-c* are located. Next, the user should execute the X utility named *xrdb* to load the X resource information into the X server database (e.g. **xrdb -load Main-c**). Finally, to execute the test pattern generation application, simply type **main-c** and <return>. The application should begin execution and the full display area should be filled with the default Grid pattern.

6. X Resource File

The *Main-c* file contains X resource definition information used to provide an easily customizable look and feel to the user interface. The file contains font and color settings for the user interface. To use the X resource settings, simply type **xrdb -load Main-c** at the UNIX prompt. The intent of this file is not to include every possible customizable option but rather to demonstrate how a file of this type is used. For instance, this particular file contains a font definition that prevents the text on the pop-up menu from being clipped on systems where the default font is larger than the menu was designed to handle. Although the clipping does not affect execution, it is visually more pleasing to prevent it.

7. The Interfaces

7.1 Graphical User Interface (GUI)

When the application executes, the default Grid pattern appears in a borderless X11R6/Motif window on the display. The window utilizes the maximum resolution of the screen. Depressing the right mouse button posts a pop-up main menu that allows the user to modify the current pattern, including increasing/decreasing the line width, rotating the ColorWheel, modifying the foreground color, displaying a different pattern, selectively displaying portions of certain patterns, or exiting the application. Dependent upon the currently displayed pattern, certain menu options will not be accessible and will appear *grayed out*. The menu options available to each pattern are represented below (Table 2).

Menu Option	Line Width	Rotate	FG Color Selection	Pattern	Buffer	Exit
Pattern						
Grid	X			X		X
ZigZag	X			X		X
GrayScales				X		X
ColorScales				X		X
ColorWheel		X		X		X
WhiteBorders	X			X		X
VerticalLines	X			X		X
VerticalLadder			X	X		X
HorizontalLadder			X	X		X
4X128X128Blocks				X		X
LeftRightTickMarks			X	X		X
Circle				X		X
Ellipse				X		X
FlatField			X	X		X
DIDspecial			X	X		X
DotBox				X		X
GrayScalesStereo				X	X	X
ColorScalesStereo				X	X	X
HorizontalGrayScales				X		X

Table 2. Pattern to Menu Option Mapping

The *Line Width* menu option has a submenu to either increment or decrement the line width. The *Rotate* menu option has a submenu to rotate the ColorWheel pattern either clockwise or counterclockwise. The *FG Color Selection* menu option causes a pop-up *Foreground Color Selection* window to appear on the display that contains a sample color widget, three slider bars and two buttons. The sample color widget displays the color corresponding to the current slider bar settings. Each of the slider bars represents one of the three color components: red, green & blue (RGB). Moving a slider bar changes the color component level and updates the sample color widget. The two buttons are labeled *Apply* and *Dismiss*. The *Apply* button updates the default foreground color (e.g. lines) for any pattern allowing foreground color modification. The

Dismiss button simply closes the *Foreground Color Selection* window. The *Pattern* menu option has a submenu to change the current pattern to any one of the available test patterns. The *Buffer* menu option has a submenu to selectively display the top, bottom, or both portions of the *GrayScalesStereo* or *ColorScalesStereo* patterns. Finally, the *Exit* menu option will terminate the application. The GUI was designed using ICS's BX GUI software tool. Although it is not mandatory, all modifications to the user interface **should** be accomplished via BX since it is easy to make a mistake when editing the required files directly.

7.2 Keyboard

Whenever a given pattern is displayed, specific keystrokes are recognized and predefined actions are executed. The following keys are recognized: *Up-Arrow/L/l*, *Down-Arrow/D/d*, *Left-Arrow/P/p*, *Right-Arrow/N/n*, *R/r*, *U*, and *C/c*. Please note that the "l" character is used as a separator within a group of keys. Pressing any key from the group results in a common action.

Up-Arrow/L/l

Pressing the Up-Arrow key (or alternately either the upper or lower case L key) increases the line width by 1 pixel for any pattern allowing modification. If the line width exceeds ten pixels, it is reset to one. Patterns allowing line width modification include *Grid*, *ZigZag*, *WhiteBorders* and *VerticalLines*.

Down-Arrow/D/d

Pressing the Down-Arrow key (or alternately either the upper or lower case D key) decreases the line width by one pixel for any pattern allowing modification. If the line width becomes zero, it is reset to ten. Patterns allowing line width modification include *Grid*, *ZigZag*, *WhiteBorders* and *VerticalLines*.

Left-Arrow/P/p

Pressing the Left-Arrow key (or alternately either the upper case P or lower case p key) decrements the current pattern index. If the index becomes negative, it is reset to the largest defined internal application index. The application displays the pattern whose internal application pattern number matches the new index. The internal application pattern number is a predefined, sequential number defined for each of the patterns used within the application. Internal application pattern numbers are defined using BX and are stored in the *creation-c.h* header file.

Right-Arrow/N/n

Pressing the Right-Arrow key (or alternately either the uppercase N or lower case n key) increments the current pattern index and displays the pattern whose internal application pattern number matches the new index. If the index equals the constant MAXPATTERNS, it is reset to zero, initiating a new cycle through the patterns. Internal application pattern numbers are defined in the *creation-c.h* header file generated by BX.

R/r

Pressing the R key increments the starting color index for the ColorWheel pattern. If the index exceeds the constant MAXCOLORS, it is reset to zero. The effect is a rotation of the ColorWheel.

U and C/c

Pressing the upper case U key toggles the screen capture capability. Initially, the screen capture capability is inaccessible. Once the screen capture capability has been toggled on, pressing either the upper or lower case C key will spawn a subprocess executing the *xwd* utility to capture the current pattern. By convention, the output file is named *captured_pattern_XXX.xwd* where XXX is the internal application pattern number. Internal application pattern numbers are defined in the *creation-c.h* header file generated by BX. This capability is useful for capturing screen-dumps of the patterns so that they can be used in documents and/or printed.

8. Standards-Based Software Development

The source code was developed using the C programming language and X11R6/Motif. The user interface was developed using the BX GUI tool. The user interface definition is stored in a user interface language (UIL) data file. BX allows the user to define the user interface and then generate C source code to build the application.

Although not an initial requirement, it soon became apparent that this application will be hosted on more than the Sun/Solaris platform for which it was conceived. To that end, every effort has been made to make this application standards-based so that it may be used on multiple platforms with minimal alteration.

9. Source Code

Some of the test pattern application source code was auto-generated by BX, some is a modified version of BX auto-generated code and the remainder was developed.

9.1 BX (Builder Xcessory) Generated

9.1.1 main-c.c

The *main-c.c* file contains the X Windows code to initialize the user interface and to enter the main X loop. The BX GUI tool generates most of the code contained within this file. There are numerous, well identified sections within the source code where a programmer can add code to perform application specific tasks.

9.1.2 creation-c.c

The *creation-c.c* file contains the routines to create the user interface components. The BX GUI tool generates **all** of the code contained within this file. This file should **not** be modified except via BX.

9.1.3 creation-c.h

The *creation-c.h* file contains pattern number macro definitions. The internal application pattern numbers are non-negative and sequential. They provide the programmer with a suitable substitute for referencing patterns via integer literals. These macros are used within the code as tags in a *switch* statement which handles displaying the desired pattern. These macros were defined and generated using the BX GUI tool and all modifications should be performed via the BX GUI tool.

9.1.4 callbacks-c.c

The *callbacks-c.c* file contains the callback stubs associated with the widgets specified when building the user interface. The BX GUI tool generates most of the code contained within this file. Since the callbacks are initially stubs, the programmer must add the source code to perform the expected response to the event triggering the callback. This file has been extensively modified.

9.1.5 datawall.uil

The *datawall.uil* file contains the user interface language (UIL) code that defines the layout of the user interface including the drawing area and pop-up menu. UIL code is a platform independent industry standard for defining user interfaces. This file is required for **all** modifications to the user interface. After the required changes are completed using BX, the programmer will generate new C code. The C code is recompiled and relinked to build the new executable. All code contained within the UIL file is generated by BX and should **never** be modified without using BX.

9.1.6 bxutils-c.c

The *bxutils-c.c* file contains utility routines generated by the BX GUI tool and they are used in the auto-generated code. Most of these routines aid manipulation of compound strings, pixmaps and widgets. These routines are not used outside the auto-generated code. All code contained within the *bxutils-c.c* file is generated by BX and should **never** be modified without using the BX.

9.2 Other

9.2.1 utilities.c

The *utilities.c* file contains routines for generating the graphics patterns and routines supporting the generation of patterns. This file contains the `RefreshDisplayArea()` routine which provides access to **all** patterns via a *switch* statement. The *switch* control variable is used to determine which pattern is displayed. Within the `RefreshDisplayArea()` routine, there is control code which determines which menu items are available for selection.

9.2.2 event_handlers.c

The *event_handlers.c* file contains all source code for the event handlers used by the application. Currently, there are event handlers for keyboard and mouse button events generated within the drawing area of the application. Each keyboard event decodes the keycode and maps specific keycodes into actions. See section 7 of this report for a more thorough discussion.

9.2.3 datawall.h

The *datawall.h* file contains definitions for all constants and functions **not** defined via BX. All color specifications are defined in accordance with the values used in the source code for the SGI/GL-based application.

10. Conventions

10.1 Naming

An application, such as this, is a work-in-progress. It is never truly complete since it is very probable that new patterns will periodically be added. Despite the fact that this application was developed on-the-fly, code maintenance was always a consideration.

In an attempt to minimize global information, yet support access to required data, numerous convenience functions were written to provide access to specific data items such as widget identifiers, color & grayscale information, and indices. All routines used to store or retrieve such information have function names beginning with *GetSet*. The remaining portion of the function name is some word or phrase to uniquely identify the type of information this function was intended to handle. For example, the convenience function to store and retrieve the drawing area widget is named *GetSetDrawingAreaWidget()*. All callback routines have function names with the suffix *CB*. The source code for all callback routines is located in the *callbacks-c.c* file.

10.2 Graphic

To minimize the effect of the Xlib graphics library algorithms on the displayed graphics output, most *n*-pixel width lines are drawn as *n* 1-pixel width lines rather than as a single *n*-pixel width line. Although this can add to rendering overhead somewhat, it ensures the image is rendered correctly. If the geometry of the displayed pattern appears distorted, the algorithm most assuredly can be eliminated as a potential source of the problems. It should be noted that the ZigZag pattern is an exception to this convention.

10.3 Error Handling

10.3.1 Informational

During the course of normal execution, if certain conditions or non-fatal errors occur an informational window is displayed. A message is displayed requiring user input to dismiss the

informational window. One common use of informational windows is to report attempts to access options not supported for the currently displayed pattern.

10.3.2 Fatal

Under certain circumstances, such as being unable to initialize the application colormap, an error message is displayed in the terminal window and execution halts.

11. Building Application

11.1 Makefile

A *makefile* for this application is provided for ease of recompiling after modifications. All required source code files are specified within the *makefile*. The *CC* variable within the *makefile* identifies the compiler the programmer has chosen. In our case, the SparcWorks *acc* compiler is used.

To build the application, create a work directory and copy the *makefile* and all source code to that directory. To use the *make* utility to build the application, simply set your working directory to where the source code and *makefile* are located then type **make**. Please note that slight modifications of the *makefile* may be required when rehosting the application to a new system.

11.2 Environment Variables

The UNIX environment variables required by this application are limited primarily to those required to perform the compilation and linking. The *make* utility, *acc* compiler, *xwd* utility and *xrdb* utility must be located somewhere within the PATH specification. The libraries must be located somewhere in the LD_LIBRARY_PATH specification.

12. Adding A New Pattern

The following steps describe the procedure to add new patterns to this application. A critical portion of this procedure requires experience using the BX GUI tool. It is assumed that the user has changed the working directory to the location of the source code including the *makefile* and UIL file.

1) Requirements Definition

Determine the requirements of the new pattern and choose a meaningful name. For the remainder of this procedure, let's assume you want to add a new pattern with a white border at the margins of the drawing area and blue diagonal lines connecting the corners. Let's also assume we have chosen to name the new pattern BlueDiagonals.

2) Define The User Interface

Execute the BX GUI tool binary file (i.e. *bx*) and open the UIL file named *datawall.uil*. There are several tasks to be performed using *bx*. Create an **integer** constant for the new pattern and use an **all** upper case version of the pattern name chosen in Step #1 of this procedure as the name. In our example, the new constant will be named BLUEDIAGONALS. The new constant should be set to the current value of a previously defined constant named MAXPATTERN. Increment the value of MAXPATTERN by one. Add a new button to the pattern pull-down menu. Modify the newly created button. Enter "Blue Diagonals" in the *labelString* resource and enter "PatternTypeCB(BLUEDIAGONALS)" in the *activateCallback* resource. Save the UIL file and generate the C source code. Exit from *bx*.

3) Add Routine To Generate New Pattern

Edit the *utilities.c* file to add the routine to generate the new pattern. Most patterns have the same basic requirements, therefore, it is often easiest to copy an existing routine and modify it as necessary to add the code for specific requirements. In our example, let's copy the WhiteBorders() routine and name the new routine BlueDiagonals(). Next, remove unnecessary code from the copied routine and add the new code to generate the pattern. Adding the new code requires knowledge of C, X11R6/Motif programming and Xlib functions. Finally, add a declaration for the new routine to the list of function declarations in the *datawall.h* header file.

4) Add New Case Clause to Switch

Next, modify the RefreshDrawingArea() routine to add a new *case* clause to the *switch* statement used to select the appropriate pattern to display. The new *case* clause should be something like this:

```
/* Use the constant created using bx and used in the
activateCallback resource as the tag field in the case
clause */

case BLUEDIAGONALS:
    BlueDiagonals(); /* This line executes the newly */
break;               /* created routine.          */
```

The *case* clause you would actually use may be somewhat more complex than the one shown above depending upon the nature of the new pattern. You should study this portion of the code to fully understand it prior to making any modifications.

5) Rebuild The Application

Clean up the directory prior to a rebuild by executing the *make* utility with the *clean* option (i.e. **make clean**) and then perform the rebuild by executing the **make** utility without options from the UNIX prompt. The *make* utility will rebuild the executable in accordance with the information and rules specified in the *makefile*. During the rebuild, the user can monitor progress by reviewing the output information from the utility.

13. Summary

This interactive test pattern environment has become an invaluable tool for frequent in-house video projector fine tuning and preventive maintenance actions. The test patterns have been effectively used to align and color balance the ADII video projectors in both an overlaid stereoscopic configuration and tiled DataWall configurations. In addition to providing a much more accurate alignment method, it is estimated to have reduced the workload required by more than one half. On several occasions the projector systems required contractor on-site maintenance actions. The contractor's preference of the test pattern environment over the internal projector test patterns attests to the utility and flexibility of the developed environment.

The first implementation of the Interactive DataWall at AFRL/IF prompted the need for an alignment and color balancing test environment. The test patterns provided a significantly improved method for aligning the three projectors and effectively balancing the color across the entire display area. The VerticalLines pattern tested the capacity of the display system to resolve very high-resolution images, to see if it was possible to resolve these lines at 1600 x 1200 pixels per projector. This was to gauge the benefits of investing in a frame buffer that would allow us to drive the projectors at their full capacity of 2500 x 2000 pixels. Our preliminary results revealed an inability of the display system to effectively display clear images of one-pixel width lines at a resolution of 1600 x 1200. The one-pixel width lines essentially look like a gray field when displayed on the projectors. The lines are discernable however on direct view monitors that are also connected to the computer system. It could indicate a focus problem and/or the need for some line amplifiers. Where the problem lies (the projectors, screen, cables, need for amplifiers, etc.) still needs to be determined. It did however put priorities in perspective for the DataWall project, forcing us to improve the display quality at our current resolution before setting out to create an even higher resolution display system.

The initial objective of this in-house task was to develop a Sun Microsystems-based version of the test pattern generation application to support Sun-based DataWall implementations. What resulted was an X Windows platform independent application capable of generating a number of test patterns used to align and adjust various characteristics of high resolution, tiled (e.g.

DataWall) or overlaid (e.g. stereoscopic 3D) projected displays. We chose X Windows as the development standard to maximize the portability potential given the economic constraints. The resulting application has proven very successful. This tool has been evolving as the DataWall and 3D stereoscopic displays evolve, with improvements made as system specific needs arise. Its future effectiveness will be determined as these display environments continue to mature, and the need for any additional enhancements and extensions are identified.

References

- [1] Alphonse, G. A., Lubin, J., "Psychophysical requirements for tiled large screen displays", SPIE Vol. 1664 High-Resolution Displays and Projection Systems, 1992, pp.230-240.
- [2] Biberman, L. M., Perception of Displayed Information, Plenum Press, New York, NY, 1973.
Holmes, R. E., "VideoramaTM - a new concept in juxtaposed large screen displays", SPIE Vol. 1081 Projection Display Technology, Systems, and Applications, 1989, pp.15-20.
- [3] Integrated Computer Solutions, Builder Xcessory User's Guide, 1998.
- [4] Jedrysik, P. A., Interactive Test Patterns for Tiled and Stereoscopic Displays, RL-TM-96-2, April 1996.
- [5] Jedrysik, P. A., Sweed R., and VanPelt, R. "Test Pattern Generation Software Final Report", May 1998.
- [6] Mashushi, T., Small, D., & MacNeil, R. L., "6,000 x 2,000 Display Prototype", SPIE Vol. 1664 High-Resolution Displays and Projection Systems, 1992, pp. 202-209.

**MISSION
OF
AFRL/INFORMATION DIRECTORATE (IF)**

The advancement and application of information systems science and technology for aerospace command and control and its transition to air, space, and ground systems to meet customer needs in the areas of Global Awareness, Dynamic Planning and Execution, and Global Information Exchange is the focus of this AFRL organization. The directorate's areas of investigation include a broad spectrum of information and fusion, communication, collaborative environment and modeling and simulation, defensive information warfare, and intelligent information systems technologies.